drawing methods of the JButton class 48. This invokes the platform-independent methods of Swing to paint the region of the screen originally allocated for the Button 30 in the AWT-based application.

Please replace the paragraph beginning on page 39, line 9, with the following rewritten paragraph. KP 8/08/08

Normally, Swing components are created only during the construction of the Peer object. However, in order to reproduce the behavior of the legacy AWT-based TextField control 170, JTextFieldPeer 172 must dynamically switch between the two Swing objects when the mode of use changes. This is accomplished by creating a new JTextFieldProxy 174 or JPasswordTextFieldProxy 176 object, depending on the settings of the echo character, and transferring the properties from the old to the new component. The appropriate Swing component is in effect "switched in" as a functional replacement for the AWT TextField, according to its mode of use. For example, if a legacy application is using ~~a~~ an AWT TextField control without password protection, no echo character is set. In this case, the AWTSwing JTextFieldPeer uses the Swing TextField component. Now, if the legacy application activates password protection, the AWT TextField will be assigned an echo character. When this occurs, the JTextFieldPeer creates an instance of Swing's JPasswordField component and substitutes it for the original AWT TextField control. This is done dynamically, so that each time the application changes the echo character status, the appropriate Swing replacement object (JTextField or JPasswordTextField) is created and used to replace the previous replacement object. Thus, the mode-switching capability of AWTSwing permits two Swing components to alternate as replacements for an AWT TextField component, depending on the manner in which the AWT TextField is being used by the application.

Please replace the paragraph beginning on page 39, line 26, with the following rewritten paragraph.

A flowchart representing the logic for dynamic TextField mode switching is presented in Fig. 18. Upon entry 192, the algorithm tests 178 the status of the echo character. If there has been no change in the state of the echo character, then whichever Swing control is currently in use (either JTextField or JPasswordField) is retained, and nothing needs to be done 190. On the other hand, if the echo status has changed, then it is necessary to swap the JTextField Swing component for a JPasswordField Swing component, or vice-versa. Before making this switch, the state (color, position, text, etc) of the control that is presently being displayed is saved 180. Once this information is captured, the current object can be destroyed 182, and its alternate created 184. After it is initialized 186, the previously saved state is applied to the ~~new~~ newly created Swing component, and the procedure is finish d 190.